

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. | |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине Объектно-ориентированное программирование

(наименование учебной дисциплины (модуля))

на тему: Объектно-ориентированная реализация игры «Змейка»

Автор проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С.Бережнов

(подпись) И.О.Ф.

Направление/специальность, профиль/специализация:

09.03.04 Программная инженерия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

код направления наименование направления (специальности)

наименование профиля (специализации)

Обозначение курсовой работы КР.780000.000 Группа ВПР33

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б.В. Габрельян\_\_\_\_\_\_\_

подпись (должность, И.О.Ф.)

Проект (работа) защищен(а) \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка подпись

Ростов-на-Дону  
2021



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. | |

ЗАДАНИЕ

на курсовой проект (работу)

Студент Бережнов А.С. Код 09.03.04 Группа ВПР33

Тема Объектно-ориентированная реализация игры «Змейка»

Срок представления проекта (работы) к защите «\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

Исходные данные для курсовой работы

1. Задание к выполнению курсовой работы

2. Основная литература: лекционный материал по дисциплине «Объектно-ориентированное программирование».

Содержание пояснительной записки

ВВЕДЕНИЕ:

В разделе «Введение» рассматривается актуальность игр, описывается жанр игры «Змейка».

Наименование и содержание разделов:

1. В разделе «Аналитический обзор» рассматриваются основные стандарты и принципы игры «Змейка».

2. В разделе «Алгоритмическое конструирование» рассматриваются основной алгоритм работы игры «Змейка» и алгоритм движения.

3. В разделе «Программное конструирование» приводится обоснование выбора языка и среды программирования для разработки игры «Змейка». Также описаны основные классы, используемые для реализации программного средства.

4. В разделе «Тестирование программного средства» произведено тестирование игры «Змейка» на легком уровне сложности с пошаговым объяснением действий.

ЗАКЛЮЧЕНИЕ:

Выполнены все задачи для реализации игры «Змейка», создано рабочее приложение.

Руководитель проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б. В. Габрельян

подпись, дата

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Бережнов

подпись, дата

Содержание

ВВЕДЕНИЕ………………………………………………………………………. 5

[1 Аналитический обзор поставленной задачи 6](#_Toc96042028)

[1.1 Описание игры «Змейка» 6](#_Toc96042029)

[1.2 Принципы создания пользовательского интерфейса для игры 6](#_Toc96042030)

[1.3 Постановка задачи для реализации игры «Змейка» 7](#_Toc96042031)

[1.4 Выводы по главе 7](#_Toc96042032)

[2 Алгоритмическое конструирование 8](#_Toc96042033)

[2.1 Общий алгоритм работы игры «Змейка» 8](#_Toc96042034)

[2.2 Алгоритм увеличения змейки и перемещения яблока 9](#_Toc96042035)

[2.3 Выводы по главе 10](#_Toc96042037)

[3 Программное конструирование 11](#_Toc96042038)

[3.1 Обоснование выбора средств разработки для реализации игры «Змейка» 11](#_Toc96042039)

[3.2 Выбор среды программирования 12](#_Toc96042040)

[3.3 Описание объектно-ориентированной структуры игры 13](#_Toc96042041)

[3.4 Выводы по главе 14](#_Toc96042042)

[4 Тестирование программного средства 15](#_Toc96042043)

[4.1 Описание процесса тестирования 15](#_Toc96042044)

[4.2 Выводы по главе 18](#_Toc96042045)

[Заключение 19](#_Toc96042046)

[Перечень использованных информационных ресурсов 20](#_Toc96042047)

[Приложение А UML-диаграмма классов 21](#_Toc96042048)

[Приложение Б Исходный код программного средства 22](#_Toc96042050)

Введение

С развитием технологий компьютеры все больше и больше вливаются в повседневную жизнь человека. Персональные компьютеры — это уже не просто машины для математических вычислений, их общедоступность значительно расширила область применения. И, конечно же, здесь не обошлось без игр. Индустрия игр быстро набирала обороты, стремительно развивалась, завоевываю все большую и большую популярность у подрастающего поколения. И если первые игры отличались простотой дизайна и логики, то современная компьютерная игра - очень сложная программа, дело рук большого коллектива разработчиков. Так или иначе, компьютерные игры занимают не последнее место в жизни многих современных детей.  
 Игру «Змейка» можно отнести к таким жанра, как головоломки и выживание. Головоломка (англ. Puzzle) — название жанра компьютерных игр, целью которых является решение логических задач, требующих от игрока задействования логики, стратегии и интуиции. Выживание (англ. survival sim или англ. survival game) — жанр компьютерных игр, разновидность симуляторов жизни, в которых основной целью игрока является сохранение жизни виртуального персонажа на фоне множества угрожающих ему опасностей [1].

Данная курсовая работа заключается в реализации игры «Змейка», написанной с применением объектно-ориентированного программирования на языке высокого уровня Java.

1 Аналитический обзор поставленной задачи

В данной главе рассматриваются основные стандарты и принципы игры «Змейка».

* 1. Описание игры «Змейка»

Snake (Питон, Удав, Змейка, классическая змейка и др.) — компьютерная игра, впервые появившаяся в кнопочном телефоне Nokia 6110. Разработана финским разработчиком Танели Арманто и выпущена компанией Nokia. До 1997 года уже существовали подобные игры на компьютерах с названиями Snake, Boa, Pithon, Serpent и другие, во множестве вариантов, для одного или двух игроков, на компьютерах разных моделей и игровых автоматов. Заслуга Танели Арманто только в том, что он сделал змейку для телефона Nokia.

Игрок управляет длинным, тонким существом, напоминающим змею, которое ползает по плоскости (как правило, ограниченной стенками), собирая еду (или другие предметы), избегая столкновения с собственным хвостом и краями игрового поля. Каждый раз, когда змея съедает кусок пищи, она становится длиннее, что постепенно усложняет игру [1].

Игра заканчивается, когда змея сталкивается с любой из частей своего тела или же столкнувшись с любой из стенок.

1.2 Принципы создания пользовательского интерфейса для игры

Пользовательский интерфейс — это способ взаимодействия пользователя с программой [2].

Принципы разработки интерфейса:

* естественность;
* согласованность;
* дружественность.

Пользовательский интерфейс разрабатываемой игры «Змейка» должен придерживаться перечисленных принципов.

**1.3 Постановка задачи для реализации игры «Змейка»**

Основной задачей является объектно-ориентированная реализация игры «Змейка». Игра должна содержать следующий основной функционал:

* возможность выбора уровня сложности;
* реализация начала игры по кнопке;
* возможность начать заново в случае проигрыша;
* изменение направления движения змейки по клавише;
* появление яблока в новом месте и увеличение змейки;
* проигрыш при столкновении змейки с любой из своих частей или стенками;
* увеличение счета игрока.

Для достижения поставленной цели необходимо решить следующие задачи:

* алгоритмическое конструирование;
* выбор среды программирования и среды разработки;
* разработка игры;
* тестирование игры.
  1. **Выводы по главе**

В данной главе было рассмотрено описание игры «Змейка», правила игры. Также были рассмотрены принципы создания пользовательского интерфейса и была поставлена задача.

1. **Алгоритмическое конструирование**

В данном разделе рассматриваются основные алгоритмы работы игры «Змейка»: общий алгоритм, алгоритм увеличения змейки и перемещения яблока.

**2.1 Общий алгоритм работы игры «Змейка»**

Игра «Змейка» начинается с того, что пользователь должен выбрать скорость движения змейки. После выбора игрок нажимает на кнопку «Начать игру» и игра начинается.

Изначально змейка находится в начале игры по левую сторону игрового поля, а изначальное яблоко в случайном месте на игровом поле. Когда змейка начала движение игрок клавишами движение может менять направление движения змейки. Все блоки в текстуре змеи двигаются друг за другом, а непосредственно сам поворот осуществляет текстурный блок головы.

Игра может закончится в двух случаях: если голова змеи коснётся любой из частей своего тела, или если голова змейки выйдет за пределы игрового поля по любую из сторон.



Рисунок 2.1 – Общий алгоритм работы игры «Змейка»

**2.2 Алгоритм увеличения змейки и перемещения яблока**

Когда текстурный блок головы змейки попадает на текстурный блок яблока, то происходит следующие: хвост змейки возвращается в предыдущую позицию, а на его место добавляется новый блок тела, который двигается вслед за тем блоком, который раньше был последней частью тела, хвост двигается за новым последним блоком тела. Яблоко перемещается на новое случайное место. За каждое съеденное змеей яблоко игрок получает 50 очков.

**2.3 Выводы по главе**

В данной главе был рассмотрен общий алгоритм работы игры, а также алгоритм увеличения змейки и перемещения яблока.

Все приведенные алгоритмы используются для дальнейшего проектирования модулей приложения и лежат в основе для реализации основных функций.

**3 Программное конструирование**

В данном разделе будут обоснованы выбор языка программирования, используемый для реализации программного средства, а также представлены основания выбора среды программирования. Будут определены и описаны основные классы разрабатываемого приложения.

**3.1 Обоснование выбора средств разработки для реализации игры «Змейка»**

Для реализации программного средства был взят за основу язык программирования Java и технология JavaFX.

Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle.

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года. Занимает высокие места в рейтингах популярности языков программирования.

JavaFX — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing. Платформа JavaFX конкурирует с Microsoft Silverlight, Adobe Flash и аналогичными системами [1].

Благодаря большому и удобному функционалу технологии JavaFX вместе с языком Java реализация игры «Змейка» будет являться объектно-ориентированной и программное средство будет содержать удобный и понятный интерфейс для пользователя.

**3.2 Выбор среды программирования**

Для реализации разработанного алгоритма была выбрана среда программирования IntelliJ IDEA.

IntelliJ IDEA — интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Первая версия появилась в январе 2001 года и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для рефакторинга, которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными свободными инструментами разработчиков, такими как CVS, Subversion, Apache Ant, Maven и JUnit. В феврале 2007 года разработчики IntelliJ анонсировали раннюю версию плагина для поддержки программирования на языке Ruby [1].

**3.3 Описание объектно-ориентированной структуры игры**

Для программной реализации игры «Змейка» и демонстрации еe работы были созданы следующие классы:

Класс Main: определяет точку входа в приложение. Не содержит функций.

Класс MainWindow: вызывается в классе Main. Реализует главное меню

Где можно начать игру, выйти, выбрать скорость движения при помощи передаваемых в класс PlayGroundWindow значений переменных speed\_1, speed\_2, speed\_3. Содержит в себе непосредственно весь вид главного меню.

Таблица 3.1 – Описание функций класса MainWindow.

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Входные параметры | Описание | Возвращаемое значение |
| void initialize () | нет | Определяет работу всех кнопок  и переключателей | нет |

Класс PlayGroundWindow: реализует непосредственно саму игру. Вызывается в классе MainWindow. Содержит непосредственно внешний вид игры. Содержит статическую переменную для определения скорости speed, а также переменные direction, LastDirection, LastTickDirection для сохранения и определения направления движения змеи. Переменную res для сохранения и вывода счета игрока, множество переменных X, Y для сохранения и использования каких-либо координат, AnimationTimer, который является непосредственной анимацией движения всей игры.

Таблица 3.2 – Описание функций класса PlayGroundWindow.

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Входные параметры | Описание | Возвращаемое значение |
| 1 | 2 | 3 | 4 |
| void initialize () | Нет | Определяет работу всех кнопок  и задает случайное изначальное место для яблока | Нет |
| void Move (KeyEvent event) | Событие клавиатуры, т.е. нажатая клавиша | Определяет по нажатой пользователем клавишей направление текущего движения | Нет |
| tick () throws IOExeption | Нет | Определяет непосредственно всю механику игры: движение, появление нового яблока и расширение змейки, конец игры | Нет |

3.4 Выводы по главе

В данной главе был обоснован выбор языка программирования, а также среда программирования для создаваемого программного средства. Кроме того, были описаны основные классы программного средства. В таблицах было приведено описание каждой функции класса с указанием типа входных параметров и возвращаемого значения.

4 Тестирование программного средства

В данном разделе продемонстрированы основные этапы тестирования игры «Змейка» на основе различных входных данных.

4.1 Описание процесса тестирования

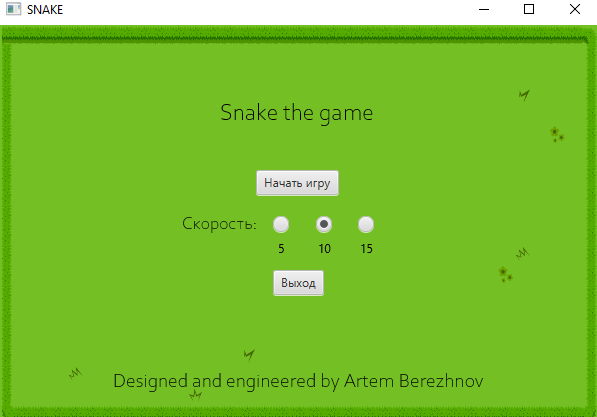
Контрольный пример №1: при начальном запуске приложения пользователю доступен выбор уровня сложности игры. Начальный запуск приложения изображен на рисунке 4.1. 

Рисунок 4.1 – Начальный запуск

Контрольный пример №2: после запуска игры «Змейка» с нужным уровнем сложности, открывается окно с игрой. Отображение приложения после выбора сложности показано на рисунке 4.2

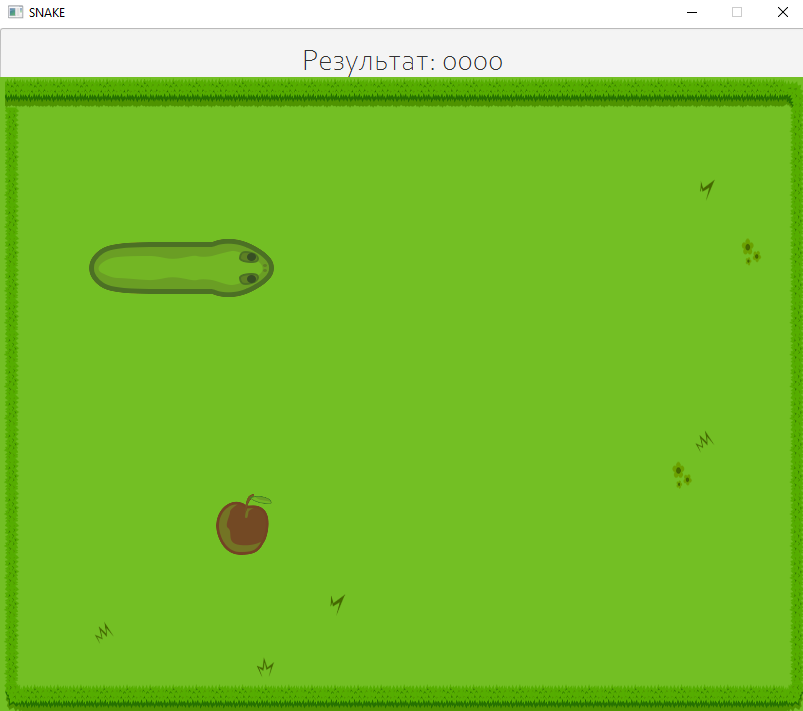


Рисунок 4.2 – Отображение приложения после выбора сложности

Контрольный пример №3: отображение приложения в ходе игры. Отображение приложения в ходе игры изображено на рисунке 4.3.

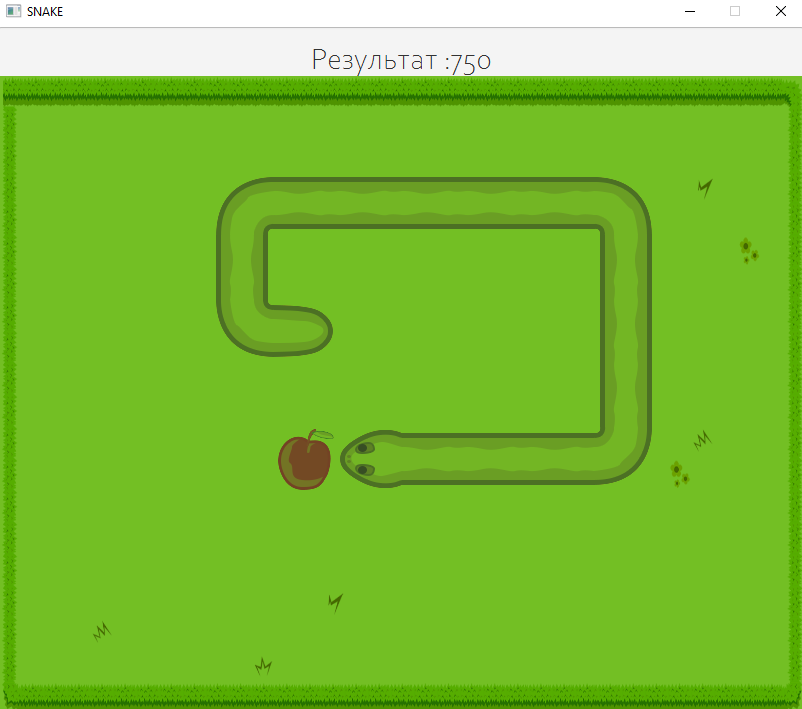


Рисунок 4.3 – Отображение приложения в ходе игры

Контрольный пример №4: после того, как игрок потратил все 3 жизни, появляется окно выбора дальнейших действий. Отображение окна выбора дальнейших действий изображено на рисунке 4.4.

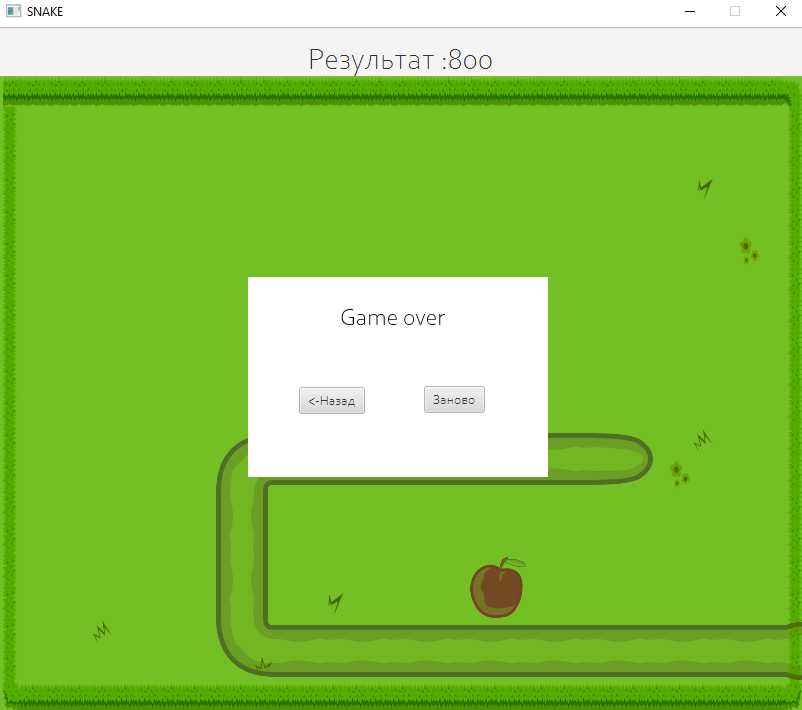


Рисунок 4.4 – Отображение окна выбора дальнейших действий

4.2 Выводы по главе

В данной главе была показана работа программного средства. Программное средство реализует все функции, которые были заданы в техническом задании. Также были решены все контрольные примеры. Программное средство работает стабильно.

Заключение

Результатом проделанной работы является программная реализация игры «Змейка».

В ходе выполнения работы был реализован алгоритм решения поставленной задачи и разработано соответствующее приложение, конечный исходный код приведен в приложении Б.

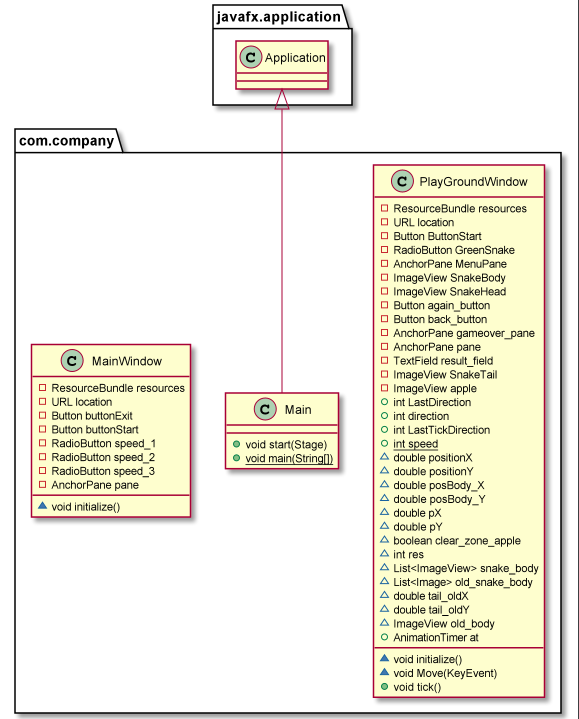
Таким образом была решена поставленная задача и выполнена проверка корректной работоспособности программного средства.

Перечень использованных информационных ресурсов

1. Википедия // [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/ (дата обращения 18.02.2022);
2. Ux-journal // Пользовательский интерфейс [Электронный ресурс] URL: [https://ux-journal.ru](https://ux-journal.ru/chto-takoe-polzovatelskij-interfejs.html%23principles%20) (дата обращения 18.02.2022);

# Приложение А

# UML-диаграмма классов



# Приложение Б

# Исходный код программного средства

Полный код программного средства находится на съемном носителе, прилагаемом к данной работе.

Листинг Б.1 – Исходный код класса Main

package com.company;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class Main extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("MainWindow.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 600, 400);  
 scene.getRoot().requestFocus();  
 stage.setTitle("SNAKE");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

Листинг Б.2 – Исходный код класса MainWindow

package com.company;  
  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.RadioButton;  
import javafx.scene.layout.AnchorPane;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
  
public class MainWindow {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private Button buttonExit;  
  
 @FXML  
 private Button buttonStart;  
  
 @FXML  
 private RadioButton speed\_1;  
  
 @FXML  
 private RadioButton speed\_2;  
  
 @FXML  
 private RadioButton speed\_3;  
  
 @FXML  
 private AnchorPane pane;  
  
 @FXML  
 void initialize() {  
  
 speed\_1.setOnMouseClicked(mouseEvent -> {  
 speed\_1.setSelected(true);  
 speed\_2.setSelected(false);  
 speed\_3.setSelected(false);  
 });  
 speed\_2.setOnMouseClicked(mouseEvent -> {  
 speed\_2.setSelected(true);  
 speed\_1.setSelected(false);  
 speed\_3.setSelected(false);  
 });  
 speed\_3.setOnMouseClicked(mouseEvent -> {  
 speed\_3.setSelected(true);  
 speed\_2.setSelected(false);  
 speed\_1.setSelected(false);  
 });  
  
  
 buttonExit.setOnAction(event ->{  
 Stage stage = (Stage) buttonExit.getScene().getWindow();  
 stage.close();  
 } );  
 buttonStart.setOnAction(event ->{  
 try {  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("PlayGroundWindow.fxml"));  
 Parent root = (Parent) fxmlLoader.load();  
  
 PlayGroundWindow pgw = fxmlLoader.getController();  
  
 if(speed\_1.isSelected()) {  
 pgw.*speed* =5;  
 }  
 if(speed\_2.isSelected()){  
 pgw.*speed* =10;  
 }  
 if(speed\_3.isSelected()) {  
 pgw.*speed* =15;  
 }  
 Stage window = (Stage) buttonStart.getScene().getWindow();  
 window.setResizable(false);  
 Scene scene = new Scene(root);  
 window.setScene(scene);  
  
 scene.getRoot().requestFocus();//ОЧЕНЬ ВАЖНЫЙ МОМЕНТ АААААААААА  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 } );  
  
 }  
}

Листинг Б.3 – Исходный код класса PlayGroundWindow

package com.company;  
  
import javafx.animation.AnimationTimer;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.RadioButton;  
import javafx.scene.control.TextField;  
import javafx.scene.effect.BlendMode;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.input.KeyEvent;  
import javafx.scene.layout.AnchorPane;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.URL;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
import java.util.ResourceBundle;  
  
public class PlayGroundWindow {  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private Button ButtonStart;  
  
 @FXML  
 private RadioButton GreenSnake;  
  
 @FXML  
 private AnchorPane MenuPane;  
  
 @FXML  
 private ImageView SnakeBody;  
  
 @FXML  
 private ImageView SnakeHead;  
 @FXML  
 private Button again\_button;  
  
 @FXML  
 private Button back\_button;  
  
 @FXML  
 private AnchorPane gameover\_pane;  
  
 @FXML  
 private AnchorPane pane;  
  
 @FXML  
 private TextField result\_field;  
  
 @FXML  
 private ImageView SnakeTail;  
  
 @FXML  
 private ImageView apple;  
  
 public int LastDirection=0;//0-right 1-left 2-up 3-down // НАПРАВЛЕНИЕ КОТОРОЕ БЫЛО В ПРОШЛОМ  
 public int direction = 0;//0-right 1-left 2-up 3-down // НАПРАВЛЕНИЕ В НАСТОЯЩЕМ  
 public int LastTickDirection=0;  
 public static int *speed* = 10;  
 double positionX=0;  
 double positionY=0;  
 double posBody\_X=0;  
 double posBody\_Y=0;  
 double pX=0;  
 double pY=0;  
 boolean clear\_zone\_apple=false;//если яблоко заспавнилось на змее;  
 int res=0000;  
  
 @FXML  
 List<ImageView> snake\_body;  
  
 @FXML  
 List<Image> old\_snake\_body;  
  
  
 double tail\_oldX=0;//координаты хвоста в прошлом тике для смещения при добавлении новой части тела  
 double tail\_oldY=0;  
  
 @FXML  
 ImageView old\_body = new ImageView();  
  
 @FXML  
 public AnimationTimer at = new AnimationTimer() {  
 long lastTick = 0;  
 public void handle(long now) {  
  
 if (lastTick == 0) {  
 lastTick = now;  
 try {  
 tick();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return;  
 }  
  
 if (now - lastTick > 1000000000 / *speed*) {  
 lastTick = now;  
 try {  
 tick();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 };  
  
 @FXML  
 void initialize() {  
 back\_button.setOnAction(event -> {  
 try {  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("MainWindow.fxml"));  
 Parent root = (Parent) fxmlLoader.load();  
 Stage window = (Stage) back\_button.getScene().getWindow();  
 window.setResizable(false);  
 Scene scene = new Scene(root);  
 window.setScene(scene);  
 scene.getRoot().requestFocus();//ОЧЕНЬ ВАЖНЫЙ МОМЕНТ АААААААААА  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 });  
  
 again\_button.setOnAction(event -> {  
 try {  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("PlayGroundWindow.fxml"));  
 Parent root = (Parent) fxmlLoader.load();  
 Stage window = (Stage) again\_button.getScene().getWindow();  
 window.setResizable(false);  
 Scene scene = new Scene(root);  
 window.setScene(scene);  
 scene.getRoot().requestFocus();//ОЧЕНЬ ВАЖНЫЙ МОМЕНТ АААААААААА  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 });  
 snake\_body = new ArrayList<>();  
 snake\_body.add(SnakeHead);  
 snake\_body.add(SnakeBody);  
 //Случайная генерация яблока в начальной позиции  
 Random random = new Random();  
 apple.setLayoutX((random.nextInt((8 - 4) + 4))\*64);  
 apple.setLayoutY((random.nextInt((8 - 4) + 4))\*64);  
 at.start();  
 }  
  
 @FXML  
 void Move(KeyEvent event) {//Смена направления движения  
 switch (event.getCode()){  
 case *W* :  
 if(direction==3 || direction==2){  
 break;  
 }  
 LastDirection=direction;  
 direction=2;  
 break;  
 case *S*:  
 if(direction==2 || direction==3){  
 break;  
 }  
 LastDirection=direction;  
 direction=3;  
 break;  
 case *A*:  
 if(direction==0 || direction==1){  
 break;  
 }  
 LastDirection=direction;  
 direction=1;  
 break;  
 case *D*:  
 if(direction==1 || direction==0){  
 break;  
 }  
 LastDirection=direction;  
 direction=0;  
 break;  
 }  
 }  
  
 public void tick() throws IOException {  
  
 tail\_oldX=SnakeTail.getLayoutX();  
 tail\_oldY=SnakeTail.getLayoutY();  
  
 positionX=snake\_body.get(0).getLayoutX();  
 positionY=snake\_body.get(0).getLayoutY();  
  
 posBody\_X=snake\_body.get(snake\_body.size()-1).getLayoutX();  
 posBody\_Y=snake\_body.get(snake\_body.size()-1).getLayoutY();  
  
  
 //Повороты  
 switch (direction){  
 case 0:  
 snake\_body.get(0).setLayoutX(snake\_body.get(0).getLayoutX()+64);  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/HeadRight.bmp");  
 Image image = new Image(input);  
 snake\_body.get(0).setImage(image);  
 if(LastDirection==2){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyDR.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastDirection==3){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyDL.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastTickDirection==direction){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyLR.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 break;  
 case 1:  
 snake\_body.get(0).setLayoutX(snake\_body.get(0).getLayoutX()-64);  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/HeadLeft.bmp");  
 image = new Image(input);  
 snake\_body.get(0).setImage(image);  
 if(LastDirection==2){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyLD.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastDirection==3){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyRD.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastTickDirection==direction){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyLR.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 break;  
 case 2:  
 snake\_body.get(0).setLayoutY(snake\_body.get(0).getLayoutY()-64);  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/HeadUp.bmp");  
 image = new Image(input);  
 snake\_body.get(0).setImage(image);  
 if(LastDirection==0){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyRD.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastDirection==1){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyDL.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastTickDirection==direction){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyTD.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 break;  
 case 3:  
 snake\_body.get(0).setLayoutY(snake\_body.get(0).getLayoutY()+64);  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/HeadDown.bmp");  
 image = new Image(input);  
 snake\_body.get(0).setImage(image);  
 if(LastDirection==0){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyLD.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastDirection==1){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyDR.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 if(LastTickDirection==direction){  
 clazz = this.getClass();  
 input = clazz.getResourceAsStream("img/GreenSnake/BodyTD.bmp");  
 image = new Image(input);  
 snake\_body.get(1).setImage(image);  
 }  
 break;  
 }  
  
 //Движение друг за другом  
 for(int i =1;i<snake\_body.size();i++) {  
 pX=snake\_body.get(i).getLayoutX();  
 pY=snake\_body.get(i).getLayoutY();  
 snake\_body.get(i).setLayoutX(positionX);  
 snake\_body.get(i).setLayoutY(positionY);  
 positionX=pX;  
 positionY=pY;  
 }  
  
  
 if(snake\_body.size()>2){  
 for (int i=2;i<snake\_body.size();i++){  
 snake\_body.get(i).setImage(old\_snake\_body.get(i-1));  
 }  
 }  
  
  
 SnakeTail.setLayoutY(posBody\_Y);  
 SnakeTail.setLayoutX(posBody\_X);  
  
  
 //Смена текстурки хвоста в зависимости от положения последней части тела  
 if (snake\_body.get(snake\_body.size()-1).getLayoutX() == SnakeTail.getLayoutX()+64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailRight.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 if (snake\_body.get(snake\_body.size()-1).getLayoutX() == SnakeTail.getLayoutX()-64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailLeft.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 if (snake\_body.get(snake\_body.size()-1).getLayoutY() == SnakeTail.getLayoutY()+64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailDown.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 if (snake\_body.get(snake\_body.size()-1).getLayoutY() == SnakeTail.getLayoutY()-64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailUp.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
  
  
  
  
 if(apple.getLayoutX()==snake\_body.get(0).getLayoutX() && apple.getLayoutY()==snake\_body.get(0).getLayoutY()) // Момент поедания змейкой яблока  
 {  
 Random random = new Random();  
  
 apple.setLayoutX((random.nextInt((8 - 4) + 4)) \* 64);  
 apple.setLayoutY((random.nextInt((8 - 4) + 4)) \* 64);  
  
 int n=0;  
 while (!clear\_zone\_apple) {  
 n=snake\_body.size();  
 for (int i = 1; i < snake\_body.size(); i++) {  
 if (apple.getLayoutX() == snake\_body.get(i).getLayoutX() && apple.getLayoutY() == snake\_body.get(i).getLayoutY()) {  
 apple.setLayoutX((random.nextInt((8 - 4) + 4)) \* 64);  
 apple.setLayoutY((random.nextInt((8 - 4) + 4)) \* 64);  
 }  
 else{  
 n-=1;  
 }  
 if(n==1){  
 clear\_zone\_apple=true;  
 }  
 }  
  
 }  
 res+=50;  
 result\_field.setText("Результат :"+res);  
 ImageView IV = new ImageView();  
 IV.setBlendMode(BlendMode.*DARKEN*);  
 IV.setImage(old\_body.getImage());  
 IV.setLayoutX(SnakeTail.getLayoutX());  
 IV.setLayoutY(SnakeTail.getLayoutY());  
 IV.setVisible(true);  
 snake\_body.add(IV);  
 pane.getChildren().add(snake\_body.get(snake\_body.size()-1));  
 SnakeTail.setLayoutX(tail\_oldX);  
 SnakeTail.setLayoutY(tail\_oldY);  
 if (snake\_body.get(snake\_body.size()-1).getLayoutX() == SnakeTail.getLayoutX()+64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailRight.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 if (snake\_body.get(snake\_body.size()-1).getLayoutX() == SnakeTail.getLayoutX()-64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailLeft.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 if (snake\_body.get(snake\_body.size()-1).getLayoutY() == SnakeTail.getLayoutY()+64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailDown.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 if (snake\_body.get(snake\_body.size()-1).getLayoutY() == SnakeTail.getLayoutY()-64){  
 Class<?> clazz = this.getClass();  
 InputStream input = clazz.getResourceAsStream("img/GreenSnake/TailUp.bmp");  
 Image image = new Image(input);  
 SnakeTail.setImage(image);  
 }  
 }  
  
 for (int i=1;i<snake\_body.size();i++){  
 if(snake\_body.get(0).getLayoutX()==snake\_body.get(i).getLayoutX() && snake\_body.get(0).getLayoutY()==snake\_body.get(i).getLayoutY()){  
 at.stop();  
 gameover\_pane.setVisible(true);  
 }  
 }  
  
 if (SnakeHead.getLayoutX() > 704 || SnakeHead.getLayoutX() < 0 || SnakeHead.getLayoutY() > 512 ||SnakeHead.getLayoutY() < 0) // Game Over  
 {  
 at.stop();  
 gameover\_pane.setVisible(true);  
 }  
  
 LastTickDirection=direction;  
 old\_body.setImage(snake\_body.get(snake\_body.size()-1).getImage());  
 old\_snake\_body = new ArrayList<>();  
 for (int i=0;i<snake\_body.size();i++){  
 old\_snake\_body.add(snake\_body.get(i).getImage());  
 }  
 }  
  
}